Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tjaša Jagodnik

Prototip učnega sistema za poučevanje aritmetike v osnovni šoli

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Igor Rožanc

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Učni sistemi so učinkovita pomoč pri izvajanju učnega procesa, vendar zahtevajo natančno poznavanje specifičnih zahtev le-tega. V diplomski nalogi prikažite razvoj učnega sistema za poučevanje aritmetike na osnovni šoli od idejne rešitve preko načrta do delujočega prototipa. Poudarek pri rešitvi naj bo na modularni sestavi, parametriziranem generiranju kvizov ter učencu prilagojenem uporabniškem vmesniku. Po predstavitvi izgleda rešitve nalogo zaključite s presojo prednosti in slabosti učnega sistema.

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Tjaša Jagodnik sem avtorica diplomskega dela z naslovom:

Prototip učnega sistema za poučevanje aritmetike v osnovni šoli (angl. Prototype of tutoring system for teaching arithmetics in primary school)

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom viš. pred. dr. Igorja Rožanca,
- so elektronska oblika diplomskega dela, naslov (slovenski, angleški), povzetek (slovenski, angleški) ter ključne besede (slovenske, angleške) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, 1. septembra 2016 Podpis avtorja:

Zahvaljujem se viš. pred. dr. Igorju Rožancu za pomoč pri izdelavi diplomskega dela. Zahvala gre tudi vsem bližnjim za pomoč in podporo med študijem.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
ACID	Atomicity, Consistency, Isolation, Durability	atomarnost, konsistentnost, izolacija, trajnost
AJAX	Asynchronous JavaScript and XML	asinhroni JavaScript in XML
API	Application Programming Interface	aplikacijski programski vmesnik
CSS	Cascading Style Sheets	kaskadne slogovne pole
EJB	Enterprise Java Beans	poslovna zrna Java
GWT	Google Web Toolkit	skupina spletno orodij Google
HTML	Hyper Text Markup Language	jezik za označevanje nadbesedila
ITS	Intelligent Tutoring System	inteligentni učni sistem
JAR	Java Archive	arhiv Java
Java EE	Java Enterprise Edition	standardna različica Java
Java SE	Java Standard Edition	poslovna različica Java
JMS	Java Message Service	sporočilna storitev Java
LMS	Learning Management System	sistem za upravljanje učenja
MOOC	Massive Open Online Courses	množični odprti spletni tečaji
ORDBMS	Object-Relational Database Management System	Sistem za upravljanje objektno-relacijskih podatkovnih baz
RPC	Remote Procedure Calls	klici oddaljenih procedur
SVG	Scalable Vector Graphics	skalabilna vektorska grafika

XML	Extensible Markup Language	razširljivi označevalni jezik
XUL	XML User Interface Language	razširljivi označevalni jezik za grafične uporabniške vmesnike

Povzetek

Naslov: Prototip učnega sistema za poučevanje aritmetike v osnovni šoli

Namen diplomske naloge je prikazati razvoj učnega sistema za poučevanje aritmetike v osnovni šoli. Razviti učni sistem predstavlja uporaben pripomoček pri izvajanju učnega procesa. Sistem implementira principe učnih sistemov, kot so upravljanje in posredovanje učnih vsebin, izvajanje preverjanja znanja ter pregled rezultatov. Glavna funkcija sistema je zagotovo avtomatsko generiranje učnih vsebin, kar učitelju pomaga k hitrejšemu izvajanju izobraževalnega procesa. Učne vsebine se zgenerirajo na podlagi parametrov, ki jih določi učitelj. Učni sistem je zasnovan na trinivojski arhitekturi, ki zajema enega ali več odjemalcev, aplikacijski in podatkovni strežnik. Razviti prototip predstavlja izhodišče za razvoj večjega učnega sistema.

Ključne besede: učni sistem, sistem za upravljanje učenja, generiranje nalog, avtomatsko preverjanje znanja

Abstract

Title: Prototype of tutoring system for teaching arithmetics in primary school

The goal of the thesis is to represent the development of the tutoring system for the teaching of arithmetic in primary school. The developed tutoring system is a useful tool in the process of teaching arithmetic. This system implements many of tutoring principles, such as the management and the transmission of teaching contents, the examination and review of the exams. The main function of this tutoring system is definitely the automatic generation of teaching contents, which makes tutoring faster and is less time-consuming for teachers. Teaching contents are generated on the basis of the parameters provided by a teacher. The architecture of the system consists of one or more clients, application server and database server. The developed prototype represents a base for a major tutoring system.

Keywords: tutoring system, learning management system, course generation, automatic examination

Poglavje 1

Uvod

Uporaba računalniških sistemov je v vsakdanjem življenju vse bolj pogosta. Njihov namen je v prvi vrsti avtomatizacija in hitrejše opravljanje pogostih opravil. Razvoj in razširjenost spletnih tehnologij sta omogočila uporabo teh aplikacij širši množici, hkrati pa je dostop do aplikacij postal možen iz različnih lokacij.

Tudi na področju poučevanja in izobraževanja si lahko pomagamo z računalniki. Razvitih je že kar nekaj učnih sistemov (angl. tutoring systems), ki zajemajo naslednje principe [1]:

- posredovanje učnih vsebin,
- izvajanje preverjanj,
- pregled rezultatov.

Med učne sistema uvrščamo tudi sisteme za upravljanje učenja (angl. Learning Management System - LMS) [3]. LMS ponujajo različne funkcionalnosti za upravljanje učenja, kot sta sestavljanje nalog in preverjanje znanja. Pri pregledu obstoječih platform se je izkazalo, da večina ne ponuja tudi generiranja učnih vsebin. Učitelj mora tako vse naloge ročno sestaviti in jih vnesti v sistem, kar je lahko zamudno, vsi učenci pa bodo reševali iste naloge.

Cilj diplomskega dela je izdelati spletno aplikacijo, ki je namenjena utrjevanju matematičnih (aritmetičnih) vsebin v nižjih razredih osnovnih šol. Gre za enostaven primer učnega okolja, ki zajema sistem za upravljanje učenja, pregled

učnih vsebin, reševanje nalog, avtomatsko generiranje nalog in pregled rezultatov. Pomemben je tudi uporabniški vmesnik, saj mora biti zanimiv za ciljno skupino uporabnikov, ki so v našem primeru učenci nižjih razredov osnovnih šol.

Spletne aplikacije so primerne za realizacijo tovstnih sistemov, saj omogočajo registracijo in posledično prijavo uporabnikov v sistem. Poleg tega nam številne spletne tehnologije ponujajo širok nabor možnosti za izboljšavo grafične podobe uporabniškega vmesnika. Na ta način lahko uporabniški vmesnik prilagodimo specifičnim okoljem in njihovim uporabnikom.

V diplomskem delu je predstavljen razvoj naše aplikacije. V prvem poglavju so najprej predstavljeni učni sistemi in že obstoječe platforme učnih okolij. Tu so predstavljene tudi tehnologije, ki smo jih uporabili za našo implementacijo. Sledi opis zahtev, ki smo si jih postavili pred samo implementacijo. V tretjem poglavju je opisan načrt podatkovne baze in arhitekture aplikacije. Sledil opis razvoja aplikacije in naše učinkovite rešitve ter analiza sistema. V zadnjem poglavju so navedene še sklepne ugotovitve ter možnosti za nadaljnji razvoj in izboljšave.

Poglavje 2

Pregled področja

Okolja, ki so namenjena izobraževanju, so vse bolj pomembna v vsakdanjem življenju. Ključna značilnost učnih sistemov, ki uporabnika motivira za nadaljnje izobraževanje in uporabo sistema, je predvsem individualnost posameznika oziroma prilagajanje in usmerjanje vsebin posamezniku [2]. Ta omogoča, da uporabnik dostopa do vsebin, ki so zanj najbolj primerne in zanimive, samo prilagajanje pa omogočajo prilagodljiva učna okolja (angl. adaptive learning environment).

2.1 E-izobraževanje

Pojem e-izobraževanje, tudi e-učenje (angl. e-learning) označuje učna okolja oziroma orodja za učenje [1]. Omogoča učenje na daljavo, kar pomeni, da je učenje postalo neodvisno od prostora in časa. Danes se večina učnih orodij in sistemov nahaja na spletu in so zato tudi lahko dostopni. E-učenje predstavlja alternativo tradicionalnemu učenju, saj individualno izobraževanje poteka hitreje, je cenejše in prinaša boljše rezultate. Omogoča tudi deljenje učnega gradiva v različnih formatih, tečaje v živo, komunikacijo z učitelji in učenci preko forumov in diskusij.

Najpomembnejša tehnologija v vsakem sistemu za e-izobraževanje je zagotovo komunikacijska. Komunikacijske tehnologije omogočajo uporabo elektronske pošte, forume, socialna omrežja in klepete znotraj sistema, tako med učenci kot učitelji. Sistem vsebuje tudi druge tehnologije, ki so pripomogle k izboljšanju tra-

dicionalnega poučevanja, npr. uporaba mikrofona, konferenčnih klicev, deljenja zaslona, virutalnih učilnic.

2.1.1 Sistemi za upravljanje učenja

Sistem za upravljanje učenja (angl. learning management system - LMS) je sistem, ki je namenjen planiranju, generiranju in izvajanju učnega procesa [3]. Sistemi za upravljanje učenja so prilagodljive spletne platforme, ki ponujajo centralizirano in avtomatizirano administracijo učnega procesa ter generiranje in hitro dostavo učne vsebine. Sistemi podpirajo prenosljivost in sledijo standardom izobraževanja, prilagajajo vsebino in stremijo k ponovni uporabi znanja (angl. knowledge reuse). Kot sisteme za upravljanje učenja označujemo sisteme za upravljanje izobraževanja in usposabljanja, kot tudi programsko opremo, ki na spletu ponuja reševanje različnih tečajev.
Uporaba sistemov za upravljanje učenja je v vsakdanjem življenju vse bolj pogosta. V mnogih večjih podjetjih jih uporabljajo za vodenje evidence usposabljanj zaposlenih tako, da beležijo njihovo registracijo in udeležbo na izobraževanjih. Sistemi se pogosto uporabljajo tudi v akademskem okolju. Glavne značilnosti sistemov v izobraževalnih ustanovah je upravljanje uporabnikov, vlog in tečajev. Sledi pregled učnega procesa, ki je definiran kot pot aktivnosti, skozi katero gre učenec in postopoma pridobiva znanje. Pomembne funkcionalnosti so tudi obveščanje učenca, ocenjevanje učenca in pregled rezultatov. Večina sistemov je spletnih, kar učiteljem oziroma administratorjem tečajev omogoča lažjo administracijo izobraževanja, učencem pa lažji dostop do samih učnih vsebin.
Slabost sistemov za upravljanje učenja je ta, da so statični in se ne prilagajajo uporabniku sistema kot inteligentni učni sistemi.

Moodle

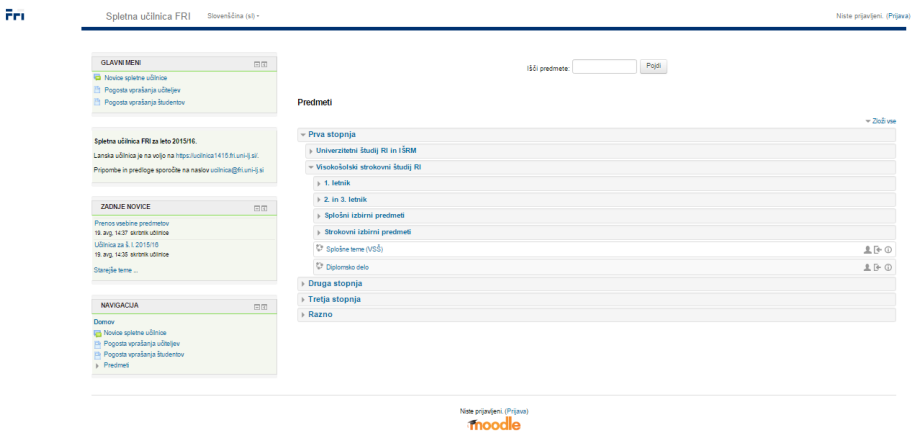
Sistem za upravljanje učenja Moodle je odprtokodna platforma, ki ponuja postavitev lastnega učnega okolja [4]. Moodle omogoča tri vloge uporabnikov:

- učitelj,

- učenec,

- administrator.

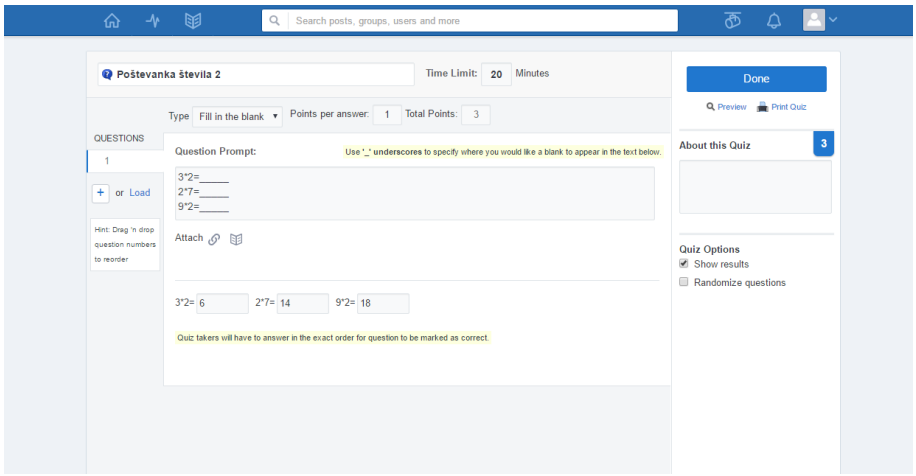
Najpomembnejše funkcionalnosti bomo tudi podrobneje predstavili [5].
Orodja za različne aktivnosti so na voljo vsem uporabnikom okolja. Učitelj lahko med uporabnike posreduje različne vrste učnih gradiv, kot so tekstovne datoteke, slikovna in video gradiva. Učitelj lahko ustvari kvize ali naloge, učencu pa je omogočeno njihovo reševanje. Na voljo so tudi orodja za komunikacijo med učenci in učitelji, kot sta klepet in forum. Primer spletne učilnice, ki uporablja platformo Moodle je prikazan na sliki 2.1.
Pomembna učiteljeva funkcionalnost je ocenjevanje. Ta poteka s pomočo kvizov, nalog, anket ...
Platforma ima možnost urejanja izgleda in jo lahko uporabljamo na različnih napravah. Prevedena je v številne jezike. Ponuja vklop obvestil o novih objavah, nalogah in pa tudi možnost pošiljanja sporočil.



Slika 2.1: Primer spletne učilnice FRI, ki uporablja platformo Moodle

Edmodo

Edmodo je brezplačna platforma za upravljanje zanja, namenjena za uporabo v osnovnih šolah [6]. Platforma ponuja dva tipa uporabnikov, to sta učitelj in učenec. Tu skrbnika ni, saj aplikacije ni potrebno nameščati na strežnik, ampak se je dostop potrebno samo registrirati.
Aplikacija predstavlja socialno okolje, ki povezuje učitelje in učence. Med seboj lahko sodelujejo ter si delijo učna gradiva in aplikacije. Platforma ponuja ustvarjanje in reševanje domačih nalog, možnost ocenjevanja in komunikacijo med uporabniki v skupinah. Primer sestavljanja kviza je prikazan na sliki 2.2. Učenci so za uspešno reaševanje nagrajeni. Z aktivno uporabo aplikacije zbirajo značke, ki so nagrada za napredek in dobro ocenjene naloge.



Slika 2.2: Primer sestavljanja kviza v platformi Edmodo

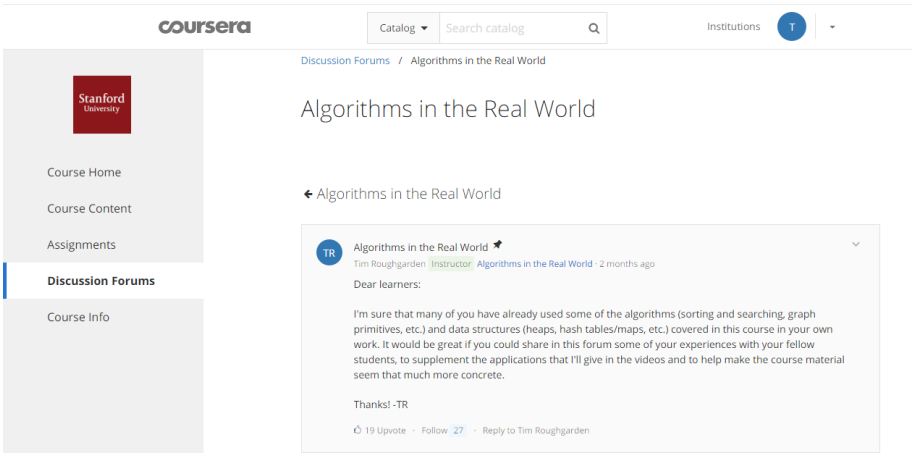
2.1.2 Množični odprti spletni tečaji

Množični odprti spletni tečaji (angl. Massive Open Online Courses - MOOC) predstavljajo novo smernico na področju e-izobraževanja, saj so brezplačni in prosto dosegljivi na spletu. Poleg tradicionalnih učnih pristopov, kot so posredovanje

video posnetkov in tekstovnega gradiva ter reševanje nalog, MOOC ponujajo tudi forum. Tu imajo učenci in učtelji možnost razprave na določeno temo. Pomembna lasnost tečajev MOOC je predvsem individualna obravnava učenca. Učenci so si med seboj enakovredni, ne glede na njihovo predznanje [7].

Coursera

Coursera je platforma, ki učiteljem ponuja možnost ustvarjanja tečajev in je brezplačna tako za učitelje kot za učence. Prednost platforme je ta, da omogoča predstavitev znanja na različne načine [8]. Ponuja interaktivne video učne vsebine, v katere je mogoče vključiti kvize in različne naloge. Coursera ima možnost deljenja tudi tekstovnih učnih vsebin v obliki učbenikov, predstavitev in zapiskov. Platforma omogoča tudi reševanje nalog. Te se delijo v tri kategorije: kvizi, programerske naloge in pa medsebojne naloge (angl. peer assessments). Slednje omogočajo učencem, da ocenjujejo naloge drugih učencev. Ponuja tudi tehnologije za komunikacijo med učenci in učitelji. Primer foruma za diskusijo je prikazan na sliki 2.3.



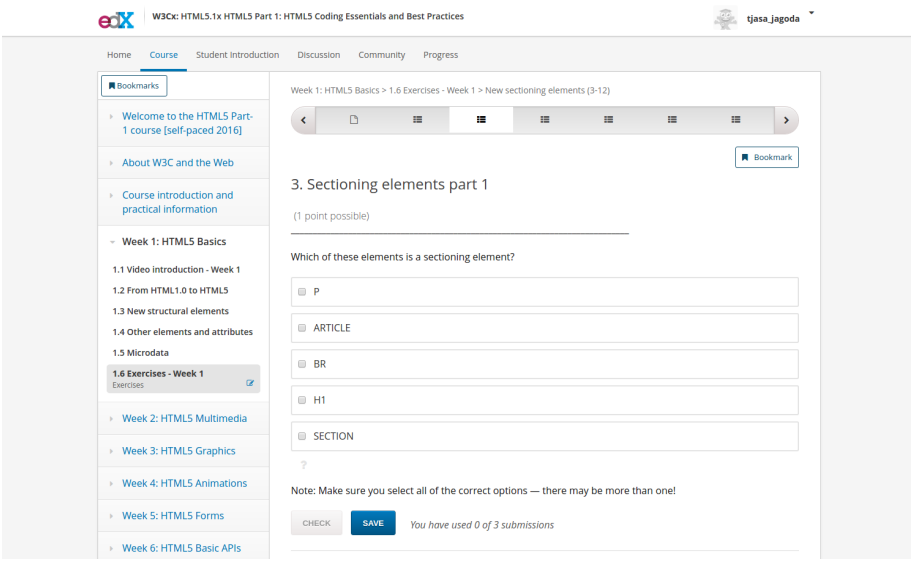
Slika 2.3: Coursera: primer foruma za diskusijo

Coursera ponuja preko 1700 tečajev iz različnih področij (matematika, računalništvo, podjetništvo, jeziki ...), ki so pripravljeni s strani priznanih univerz. Za

vsak tečaj so navedene tudi zahteve o predznanju iz izbranega področja. Po zaključku reševanja tečaja uporabnik prejme certifikat oziroma priznanje o opravljenem tečaju.

EdX

EdX je platforma za ustvarjanje množičnih odprtih spletnih tečajev [9]. Primer kviza za reševanje je prikazan na sliki 2.4. Tako kot Coursera tudi edX ponuja možnost video učnih vsebin z interaktivno nalogo, ki učencem nudi učenje vsebin iz video posnetkov. Tudi tu je mogoče deliti besedilno gradivo in sodelovati v razpravah na forumih. Platforma edX omogoča tudi delo v spletnih laboratorijih. Platforma edX ponuja več kot 1000 tečajev, ki so prav tako pripravljeni s strani univerz. Po zaključenem tečaju uporabnik prejme priznanje o opravljenem tečaju.

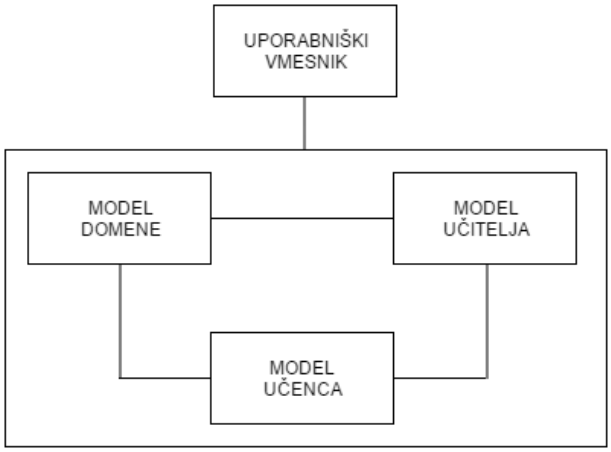


Slika 2.4: edX: primer kviza za reševanje

2.2 Inteligentni učni sistemi

Inteligentni učni sistem (angl. Intelligent Tutoring System - ITS) je učno okolje in didaktični pripomoček, ki vsebuje komponente umetne inteligence [10]. Sistem spremlja in beleži aktivnosti učenca ter se dinamično odziva na njegove dejavnosti skozi ves proces izobraževanja. Na podlagi informacij o uspešnosti reševanja, sistem sklepa o boljših in slabših področijh učenca in predlaga dodatne naloge.
Arhitekturo inteligentnega učnega sistema (slika 2.5) sestavljajo:

- **model domene** (angl. domain model),
- **model učitelja** (angl. tutoring model),
- **model učenca** (angl. student model),
- **uporabniški vmesnik** (angl. user interface).



Slika 2.5: Arhitektura inteligentnih učnih sistemov

Model domene vsebuje zbirko podatkov oziroma znanja iz danega področja. Predstavlja ključen del učnega sistema, saj ga sistem uporablja za reševanje nalog in ocenjevanje.
Model učitelja hrani učni načrt in cilj, ki naj bi ga učenci dosegli med procesom izobraževanja. Strategijo učenja določi iz podatkov modela učitelja in iz podatkov

modela domene. Prilagaja strategijo učenja potrebam učenca. Ukrepa v primerih, ko ima učenec težave.
Model učenca hrani podatke o predznanju učenca in o njegovem osvojenem znanju. V modelu učenca se hranijo tudi informacije o napakah učenca. Tu se opravi tudi analiza učenca.
Uporabniški vmesnik je potreben za interakcijo uporabnika z zalednim sistemom.

2.3 Uporabljene tehnologije

2.3.1 Java Enterprise Edition

Platforma Java Enterprise Edition (Java EE) je poslovna različica platforme Java in je izpeljana iz standardne različice Java Standard Edition (Java SE) [11]. Platforma je bila razvita za implementacijo integriranih, večnivojskih, razširljivih in varnih spletnih aplikacij. Platforma se največkrat uporablja za razvoj poslovnih aplikacij. Te zagotavljajo poslovno logiko podjetja. Ker morajo biti poslovne aplikacije varne in zanesljive, hitro postanejo tudi zahtevne in jih je težje vzdrževati. Java EE zmanjšuje zahtevnost razvoja poslovne aplikacije tako, da ponuja razvojni model, vmesnik API in delovno okolje (angl. runtime environment).

2.3.2 Enterprise JavaBeans

Enterprise JavaBeans (EJB) je tehnologija platforme Java EE, ki omogoča implelentacijo poslovnega nivoja aplikacije. Zrna (angl. beans) delimo v tri kategorije [12]:

- sejna zrna (angl. session beans),
- entitete (angl. entity beans),
- sporočilna zrna (angl. message driven beans).

Sejna zrna shranjujejo podatke o določenemu uporabniku za posamezno sejo. Delimo jih na sejna zrna s stanjem (angl. stateful) ali brez stanja (angl. stateless).

Sporočilna zrna se uporabljajo za interakcijo s sporočilnim sistemom JMS.
Entitete predstavljajo objekte, za hranjenje podatkov. Uporabljajo se za dostop do podatkov iz baze.

2.3.3 WildFly

Wildfly je odprto kodni aplikacijski strežnik, ki temelji na Java EE tehnologiji [13]. Zagon strežnika je zelo optimiziran, saj se pomembne storitve zaženejo paralelno, ostale pa le po potrebi. Vizija aplikacijskega strežnika Wildfly temelji na povezljivosti, odzivnosti in zmožnosti skaliranja. Osnovne storitve so bile zgrajene z namenom čim manjše porabe delovnega pomnilnika. Zaradi majhne porabe virov je primeren za poganjanje tudi na manj zmogljivih računalnikih. Implementira zadnje Java EE standarde ter modularno nalaga arhive JAR.

2.3.4 PostgreSQL

PostgreSQL je odprtokodni objektno-relacijski sistem za upravljanje podatkovnih baz (angl. object-relational database management system - ORDBMS), ki ga razvija PostgreSQL Global Development Group [14]. Gre za zelo zanesljiv in napreden sistem, saj vključuje tudi funkcionalnosti, ki jih ponujajo predvsem komercialni produkti. Skladen je s standardom SQL:2011 in skupkom lastnosti imenovanim ACID (angl. Atomicity, Consistency, Isolation, Durability), ki se nanašajo na podatkovne transakcije. PostgreSQL je sposoben izvajanja zahtevnih poizvedb s pomočjo metod za indeksiranje, ki jih ostale baze nimajo. Omogoča tudi veliko število različnih podatkovnih tipov, napredno iskanje nad tekstovnimi polji ter posodobljive in matrializirane poglede.

2.3.5 Google Web Toolkit

Google Web Toolkit (GWT) je odprtokondno orodje namenjeno razvoju in optimizaciji zahtevnih spletnih aplikaciji [15]. Paket vključuje različne aplikacijske programske vmesnike Java (angl. Java APIs), knjižnice, prevajalnik in strežnik

namenjen razvoju. Orodje GWT omogoča pisanje aplikacij AJAX v Javi, ki delujejo na strani odjemalca (angl. client-side). Izvorna koda se potem prevede (angl. compile) v zelo optimizirano JavaScript kodo, ki deluje na vseh glavnih brskalnikih, tudi mobilnih. Razhroščevanje aplikacij lahko izvajamo znotraj razvojnega okolja, tako kot ostale namizne aplikacije (angl. desktop applications), ali pa znotraj samega brskalnika. GWT izvaja tudi celovito (angl. comprehensive) optimizacijo izvorne kode in njeno segmentacijo v večje število JavaScript drobcev (angl. fragment), kar omogoča implementacijo optimiziranih spletnih aplikacij.

2.3.6 Kaskadne slogovne pole

Kaskadne slogovne pole (angl. Cascading Style Sheets - CSS) so slogovni jezik, ki se uporablja za opis prezentacije dokumenta, ki je napisan v označevalnem jeziku (angl. markup language) [16]. Kaskadne slogovne pole so lahko dodane dokumentom XML, SVG in XUL. Največkrat pa se uporabljajo za predstavitev strani HTML. CSS je temeljna tehnologija spleta. Ustvarjen je bil za ločitev vsebine dokumenta od predstavitve dokumenta. Takšna razdelitev dokumenta poveča dostopnost vsebine, zagotovi večjo fleksibilnost in kontrolo pri specificiranju predstavitve strani. Zmanjšuje zahtevnost in ponavljanja v strukturi vsebine. Omogoča nam tudi, da prikažemo isto vsebino na vizualno različne načine.

Poglavje 3

Zasnova učnega sistema

3.1 Zahteve

Razvoj učnega sistema je potekal v več korakih. Najprej smo določili vloge oziroma tipe uporabnikov sistema:

- učenec,
- učitelj,
- skrbnik.

Za posamezno vlogo smo tako definirali funkcionalne kot nefunkcionalne zahteve. Na podlagi zahtev smo strukturirali podatkovni model z vsemi potrebnimi tabelami. Sledila je zasnova arhitekture. Odločili smo se za trinivojsko arhitekturo.

3.1.1 Funkcionalne zahteve

Funkcionalne zahteve smo razdelili glede na vloge uporabnikov v sistemu. Zahtevi, ki sta skupni vsem uporabnikom, sta prijava in odjava.
Prijava poteka z uporabniškim imenom in geslom. Po uspešni prijavi ima uporabnik dostop do funkcionalnosti, ki so mu dodeljene glede na njegovo vlogo v sistemu.

Odjava iz sistema je omogočena vsem uporabnikom. Odjava se izvede tudi ob zaprtju brskalnika.

Učenec

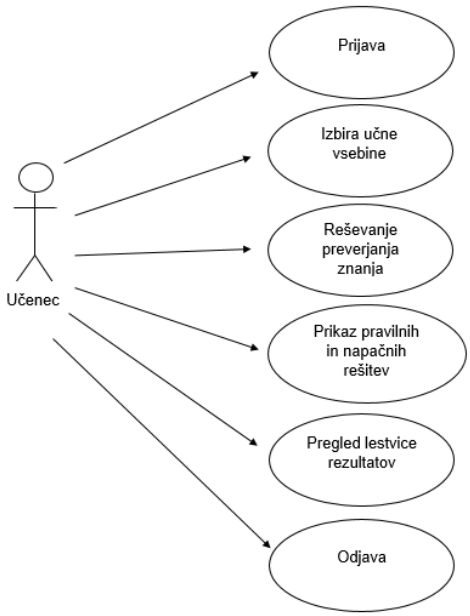
Najpomembnejša učenčeva funkcionalnost je možnost utrjevanje znanja. Slika 3.1 prikazuje diagram primerov uporabe za učenca. Pregled njegovih funkcionalnosti je naslednji:

- prosta izbira učne vsebine med vsemi vsebinami, za katere ima omogočen dostop,
- reševanje preverjanja znanja za izbrano učno vsebino. Pred pričetkom reševanja se v ozadju avtomatsko zgenerirajo naloge za izbrano učno vsebino,
- pregled rezultatov preverjanj znanja po zaključku reševanja ter pregled pravilnih in napačnih odgovorov,
- pregled vseh rezultatov izbranega preverjanja znanja in primerjava rezultata učenca na lestvici.

Učitelj

Vloga učitelja je najpomembnejša v sistemu. Učitelj namreč pripravi učne vsebine in ima pregled nad rezultati. Primeri uporabe so predstavljeni na diagramu na sliki 3.2. Učiteljeve funkcionalnosti so naslednje:

- dodajanje novega uporabnika z vlogo učenca,
- dodajanje novih učnih vsebin (definiranje posameznih stopenj)
  - stopnjo določajo naslednji parametri: čas reševanja, pogoj za napredovanje, operacija, razpon števil, število nalog istega tipa, točkovno ovrednotenje nalog,
- urejanje obstoječih učnih vsebin (urejanje, brisanje in dodajanje novih stopenj),



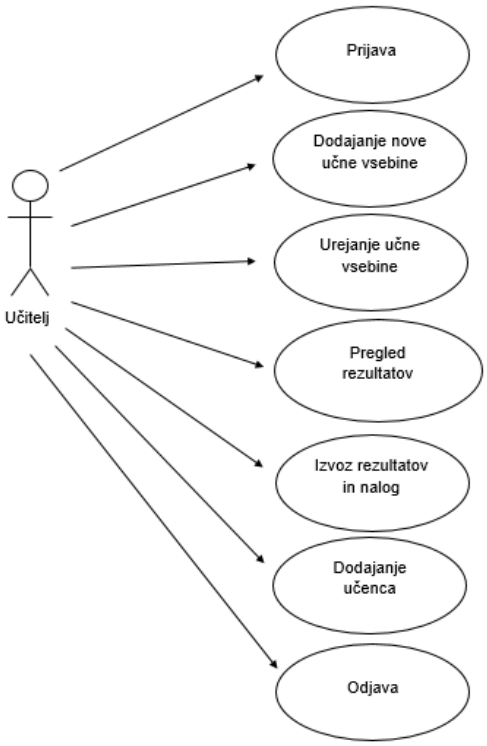
Slika 3.1: Diagram primerov uporabe za učenca

- statistika in prikaz rezultatov z grafom in tabelo za:
  - posameznega učenca po izbranih učnih vsebinah, ki jih je učenec reševal,
  - posamezno stopnjo, brez določitve učenca,
- Izvoz
  - izbranih rezultatov v formatu `pdf`,
  - izbranih učnih vsebin v formatu `pdf`.

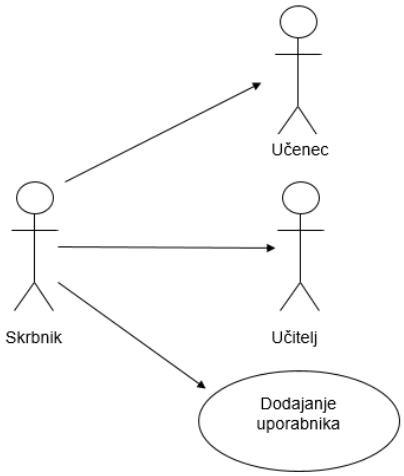
Skrbnik

Skrbnik ima največ pravic v sistemu. Lahko je v vlogi učenca ali učitelja. Slika 3.3 prikazuje diagram njegovih primerov uporabe. Dodatna skrbnikova funkcionalnost je:

- dodajanje novega uporabnika, ki je lahko učitelj ali učenec.



Slika 3.2: Diagram primerov uporabe za učitelja



Slika 3.3: Diagram primerov uporabe za skrbnika

3.1.2 Nefunkcionalne zahteve

Grafični uporabniški vmesnik

Pri definiranju ustreznega grafičnega uporabniškega vmesnika se je potrebno osredotočiti na ciljno skupino uporabnikov. Učenci, ki bodo uporabljali aplikacijo, obiskujejo nižje razrede osnovne šole. Zato mora biti uporabniški vmesnik zanimiv in privlačen, spominjati mora na računalniško igro, hkrati pa mora biti učinkovit, prijazen in za uporabnika enostaven. Uporabniški vmesnik za učitelja in skrbnika naj bo preprost in minimalističen. Vse informacije so predstavljene zelo nazorno in jasno.

Hitrost

Potrebno se je osredotočiti tudi na odzivnost aplikacije. Omogočati mora sočasen dostop večim uporabnikom, hkrati pa mora biti hitra.

Varnost

Aplikacija bo omogočala prijavo v sistem in bo hranila podatke o učencih, zato se je potrebno osredotočiti tudi na varnost. Potrebno je implementirati učinkovit generator uporabniških imen in gesel ter poskrbeti, da so gesla kriptirana.

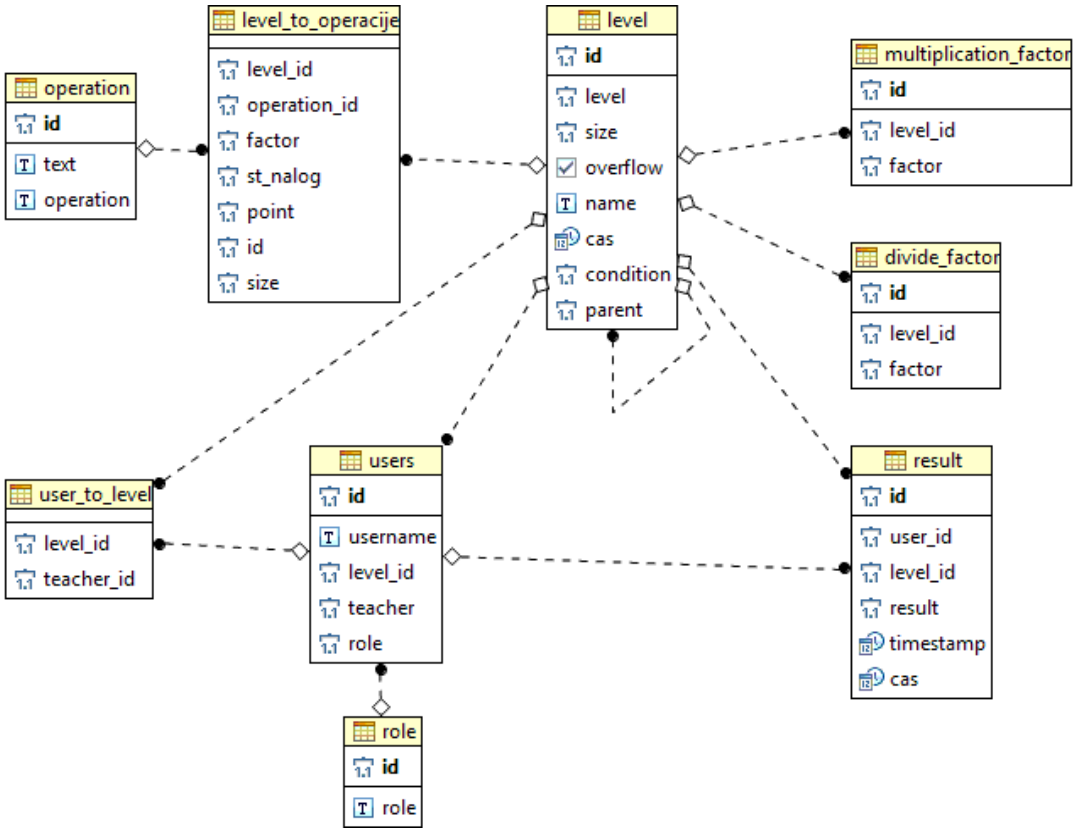
3.2 Načrt

3.2.1 Podatkovni model

Shema podatkovne baze je relacijska in vsebuje 9 tabel. ER diagram podatkovnega modela je prikazana na sliki 3.4.

Tabele, potrebne za delovanje aplikacije, so `Level`, `Result`, `Operation`, `User`, `Role`, `Multiplication_factor`, `Divide_factor`, `User_to_level` in `Level_to_operation`. Tabela `User` hrani podatke o uporabnikih sistema, v tabeli `Role` pa se nahajajo vse vloge uporabnikov. Vsaka entiteta v tabeli `Level` predstavlja učno vsebino ali pa posamezno stopnjo učne vsebine. Tabela

`Operation` vsebuje vse operacije, za katere je mogoče ustvariti nalogo. Pri nalogah množenja in deljenja se faktorjim, s katerimi množimo ali delimo, hranijo v tabelah `Multiplication_factor` in `Divide_factor`. Med tabelama `Level` in `Operation` obstaja povezava mnogo proti mnogo (angl. many-to-many), zato je dodana povezovalna tabela `Level_to_operation`. V tabeli `Result` se nahajajo vse aktivnosti učencev. V tabeli `Level_to_user` hranimo vse učne vsebine, ki so omogočene učencu.



Slika 3.4: ER diagram podatkovnega modela

Entitete podatkovnega modela so naslednje:

- **Učna vsebina** (`Level`)
  - enoličen identifikator učne vsebine,

  - stopnja učne vsebine,
  - razpon števil za računanje,
  - možnost prehoda pri odštevanju,
  - ime učne vsebine,
  - časovna omejitev,
  - pogoj za napredovanje,
  - ali gre za krovno učno vsebino ali stopnjo,

- **Rezultat** (`Result`)
  - enoličen identifikator rezultata,
  - enoličen identifikator učenca,
  - enoličen identifikator učne vsebine,
  - rezultat reševanja preverjanja znanja,
  - čas reševanja preverjanja znanja,
  - časovni žig,

- **Operacija** (`Operation`)
  - enoličen identifikator operacije,
  - opis operacije,
  - operator ali znak operacije,

- **Uporabnik** (`User`)
  - enoličen identifikator uporabnika,
  - uporabniško ime,
  - geslo,
  - enoličen identifikator zadnje učne vsebine, za katero je učenec reševal preizkus znanja,

  - vloga uporabnika,

- **Vloga** (`Role`)
  - enoličen identifikator vloge,
  - opis vloge,

- **Faktor množenja** (`Multiplication_factor`)
  - enoličen identifikator faktorja množenja,
  - enoličen identifikator učne vsebine,
  - faktor množenja,

- **Faktor deljenja** (`Divide_factor`)
  - enoličen identifikator faktorja deljenja,
  - enoličen identifikator učne vsebine,
  - faktor deljenja,

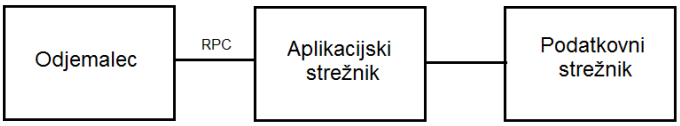
- **Učne vsebine za učenca** (`User_to_level`)
  - enoličen identifikator učne vsebine,
  - enoličen identifikator učena,

- **Naloge za učne vsebine** (`Level_to_operation`)
  - enoličen identifikator zapisa,
  - enoličen identifikator učne vsebine,
  - enoličen identifikator operacije,
  - število nalog,
  - vrednost naloge.

3.2.2 Aplikacija

Trinivojska arhitektura

Arhitektura aplikacije je trinivojska. Za takšno arhitekturo je značilno, da ima na eni stani enega ali več odjemalcev, na drugi strani pa aplikacijski strežnik [17]. Aplikacijski strežnik ima dostop in povezavo še s podatkovnim strežnikom. Značilno je, da odjemalec vsebuje samo uporabniški vmesnik, aplikacijski strežnik nosi vso logiko aplikacije, na podatkovnem strežniku pa se nahaja podatkovna baza. Model trinivojske arhitekture je predstavljen na sliki 3.5.



Slika 3.5: Prikaz trinivojske arhitekture

Za razvoj naše aplikacije smo se odločili, da za implementacijo odjemalčevega dela (angl. client-side) uporabimo tehnologijo GWT, za implementacijo strežniškega dela (angl. server-side) pa tehnologijo EJB. Za podatkovni del smo uporabili tehnologijo PostgreSQL.

Odjemalec in aplikacijski strežnik med seboj komunicirata s klici oddaljenih procedur (angl. Remote Procedure Call - RPC) [18]. Ogrodje GWT RPC omogoča izmenjavo objektov Java [19]. Klici so asinhroni (angl. asynchronous calls), kar nam omogoča vzporedno delovanje. Ko brskalnik želi dostopati do podatkov na strežniku ali jih želi shranjevati, s klicem RPC pošlje zahtevek HTTP (angl. HTTP request) na strežnik. Med procesiranjem klica RPC strežnik izvaja kodo na strežniškem delu. Mehanizem RPC poskrbi za serializacijo (angl. serialization) objektov, ki je potrebna za prenos objektov po mreži. Ob klicu RPC je potrebno specificirati metodo povratnega klica (angl. callback), ki se izvrši ob koncu klica.

Poglavje 4

Razvoj

V tem poglavju bo predstavljena implementacija naše rešitve. Opis rešitve smo razdelili po funkcionalnostih prototipa. Rešitve ključnih funkcionalnostih bomo tudi podrobnejše predstavili.

4.1 Prijava

Prijava je pomemben del spletnih aplikacij, saj uporabnika lahko identificiramo in posledničnо spremljamo le tako, da se prijavi. Aplikacijski strežnik Wildfly ponuja številne prijavne module [20]. Sami smo se odločili za `DatabaseServerLoginModule`, saj omogoča avtentikacijo (angl. authentication) in preslikavo vlog (angl. roles mapping). Uporabniška imena, gesla in vloge so shranjeni v podatkovni bazi.

Prikaz ustrezne konfiguracije strežnika:

```
1 <login-module
    code="org.jboss.security.auth.spi.DatabaseServerLoginModule"
    flag="required">
2   <module-option name="dsJndiName"
      value="java:/datasources/TutoringSystem-JPA"/>
3   <module-option name="principalsQuery" value="select password
      FROM User where username=?"/>
4   <module-option name="rolesQuery" value="SELECT r.role, 'Roles'
```

```
        FROM Roles r, User u WHERE u.username=? AND u.role = r.id"/>
    <module-option name="hashAlgorithm" value="MD5"/>
</login-module>
```

Aplikacija vsebuje eno vstopno točko za vse uporabnike (slika 4.1). Po uspešni prijavi je potrebno preveriti vlogo uporabnika, da mu dovolimo dostop do vsebine, ki pripada njegovi vlogi. V ta namen smo definirali ustrezni servlet, ki preveri vlogo uporabnika in ga ustrezno preusmeri.

Prikaz definicije prijavnega servleta:

```
@WebServlet("/login")
public class LoginServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
      if (request.isUserInRole("user_default")) {
        response.sendRedirect("student.html");

      } else if (request.isUserInRole("teacher_default")) {
        response.sendRedirect("teacher.html");

      } else if (request.isUserInRole("admin_default")) {
        response.sendRedirect("admin.html");
      }
    }

    @Override
    protected void doPost(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException,
        IOException {
      doGet(req, resp);
    }
}
```

Implementiran prijavni servlet razširja javanski servlet `HttpServlet`, zato je potrebno ponovno definirati metodi `doGet()` in `doPost()`. Metoda `doGet()` odgovarja na zahtevke GET (angl. GET requests), metoda `doPost()` pa na zahtevke POST (angl. POST requests). Iz zahteve tipa `HttpServletRequest` metoda preveri, katera vloga pripada uporabniku in preusmeri odgovor na ustrezen `html` dokument.

Slika 4.1: Vstopna točka aplikacije

4.2 Učenec

4.2.1 Izbira učne vsebine

Učenec lahko poljubno izbira med vsebinami za učenje, do katerih ima dostop. Dostop do posamezne učne vsebine mu omogoči učitelj. Za prikaz ustreznih učnih vsebin je potrebno narediti poizvedbo nad tabelo `Level_to_user`. En zapis v tabeli predstavlja učno vsebino, ki pripada učencu. Primer vsebin, ki jih ima učenec na voljo, prikazuje slika 4.2.

Slika 4.2: Vsebine, ki jih ima na voljo učenec

4.2.2 Generiranje nalog

Avtomatsko generiranje nalog se izvede, če sta izpolnjena dva pogoja. Učitelj mora predhodno definirati učno vsebino (opredeljeno v 4.3.1) in omogočiti reševanje učencem, učenec pa mora izbrati to učno vsebino za reševanje (opredeljeno v 4.2.1).

Za izbrano učno vsebino, ki je zapisana v entiteti `Level`, se najprej poiščejo vse operacije, ki pripadajo izbrani vsebini. Za vsako operacijo se zgenerira določeno število nalog. Število nalog je določil učitelj in je shranjeno v entiteti `Operation`. Preden se zgnenerirano nalogo doda na uporabniški vmesnk se preveri, ali je bila naloga že ustvarjena. Da bi se naloge podvajale je bolj verjetno v primerih, ko imamo manjši razpon števil (npr. števila do 10). Pri operacijah množenja se poišče faktor množenja, ki ga je določil učitelj, drugi faktor pa se določi naključno z uporabo funkcije `Random`. Pri operacijah deljenja se poišče deljitelj, ki ga je prav tako že določil učitelj, deljenec pa se določi naključno. Tukaj se izračuna še rezultat naloge in se shrani v objekt `Naloga`. Shrani se tudi vrednost naloge.

Zgledi generiranja nalog za podane parametre:

1. generiranje nalog za operacijo seštevanja

Parametri	**Naloge**
Operacija: Seštevanje	13 + 5
Velikost: 20	7 + 3
Število nalog: 3	16 + 1

2. generiranje nalog za operacijo odštevanja

Parametri	**Naloge**
Operacija: Odštevanje	35 - 18
Velikost: 50	41 - 25
Število nalog: 3	21 - 3

3. generiranje nalog za operacijo odštevanja

4. generiranje nalog za operacijo množenja

Parametri	**Naloge**
Operacija: Množenje	4 * 2
Velikost: 10	8 * 2
Število nalog: 4	1 * 5
Faktorji množenja: 2, 5	7 * 5

5. generiranje nalog za operacijo deljenja

Parametri	**Naloge**
Operacija: Deljenje	10 / 5
Velikost: 100	35 / 5
Število nalog: 4	60 / 10
Faktorji množenja: 5, 10	90 / 10

Primer metode, ki generira nalogo za podano stopnjo (učno vsebino) in operacijo:

```
1 public Naloga generirajNalogo(Level level, Operacije operacija) {
2   final Integer velikost = level.getSize();
3
4   Random random = new Random();
5   int prviOperator = random.nextInt(velikost - 1) + 1;
6   int drugiOperator = random.nextInt(velikost - 1) + 1;
7
```

```
8   int rezultat = Integer.MIN_VALUE;
9
10  switch (operacija.getOperacija()) {
11    case Constants.RacunskeNaloge.SESTEVANJE:
12      rezultat = prviOperator + drugiOperator;
13      break;
14    case Constants.RacunskeNaloge.ODSTEVANJE:
15      if (prviOperator == 1) {
16        drugiOperator = 1;
17      } else {
18        drugiOperator = random.nextInt(prviOperator - 1) + 1;
19      }
20      rezultat = prviOperator - drugiOperator;
21      break;
22    case Constants.RacunskeNaloge.MNOZENJE:
23      drugiOperator = getFaktorjeMnozenja(level, random);
24      rezultat = prviOperator * drugiOperator;
25      break;
26    case Constants.RacunskeNaloge.DELJENJE:
27      drugiOperator = getFaktorjeDeljenja(level, random);
28      int veckratnik = random.nextInt(velikost - 1) + 1;
29      prviOperator = drugiOperator * veckratnik;
30      rezultat = prviOperator / drugiOperator;
31      break;
32  }
33  Naloga naloga = new Naloga(prviOperator, drugiOperator,
        rezultat, operacija);
34  naloga.setTocke(operacija.getTocke());
35
36  return naloga;
37 }
```

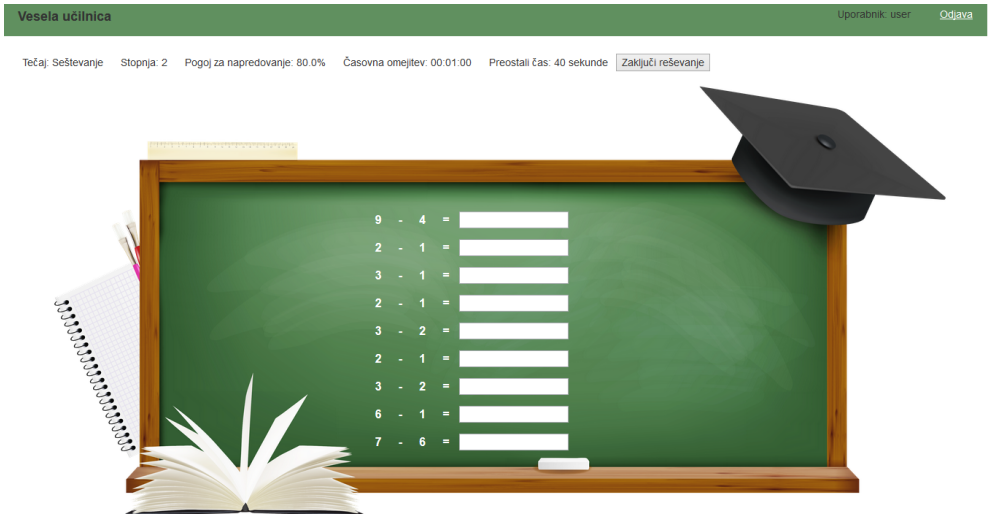
Metoda `generirajNalogo` sprejme dva parametra in vrača objekt tipa `Naloga`. Parametra sta učna vsebina (tipa `Level`) in operacija (tipa `Operation`). Iz entitete `Level` najprej dobi razpon števil s katerimi operiramo. S pomočjo funkcije `Random` poišče prvi in drugi operator. Iskanje je omejeno z razponom števil, ki se nahaja v entiteti `Level`. Metoda nato ugotovi za katero operacijo naj

zgenerira nalogo. V primeru, da je operacija seštevanje, metoda samo izračuna vsoto in se shrani v spremenljivko `rezultat`. Če gre za operacijo odštevanja, metoda najprej preveri ali je zmanjševanec enak 1. V tem primeru nastavi tudi odštevanec na vrednost 1, da prepreči negativni rezultat, sicer pa zgenerira nov odštevanec, ki se nahaja med 1 in zmanjševancem. Izračuna se še razlika in se shrani v spremenljivko `rezultat`. V primeru, da gre za operacijo množenja, metoda najprej dobi faktorje množenja. Naključno izbere enega izmed faktorjev in ga uporabi kot množitelja v nalogi. Izračuna produkt števil in jih shrani v rezultat. Če gre za operacijo deljenja, metoda najprej dobi faktorje deljenja oziroma deljitelje in enega naključno izbere. Naključno določi eno število in ga zmoži z deljiteljem, da izračuna deljenec. Na ta način si zagotovi, da so naloge vedno izračunljive.

4.2.3 Reševanje nalog

Naloge za reševanje se avtomatsko generirajo ob izbiri učne vsebine (opisano v 4.2.2). Primer izvajanja preverjanja znanja je prikazan na sliki 4.3. Čas za reševanje nalog določi učitelj. Za učne vsebine, ki imajo časovno omejitev, se nastavi odštevalnik. Ta je implementiran z GWT razredom `Timer`, kateremu je potrebno povoziti metodo `run()` in definirati svojo. V naši metodi se od trenutnega časa odšteje sekunda in se preveri, ali je čas že potekel. V tem primeru se reševanje zaključi, sicer pa se ponovno izvede metoda `run()`. Ob zaključku reševanja se rezultat shrani v podatkovno bazo.
Primer shranjevanja v bazo:

```
1 @PersistenceContext EntityManager em;
2 public void saveRezultat(Rezultat rezultat) {
3   em.persist(rezultat);
4 }
```



Slika 4.3: Primer izvajanja preverjanja znanja

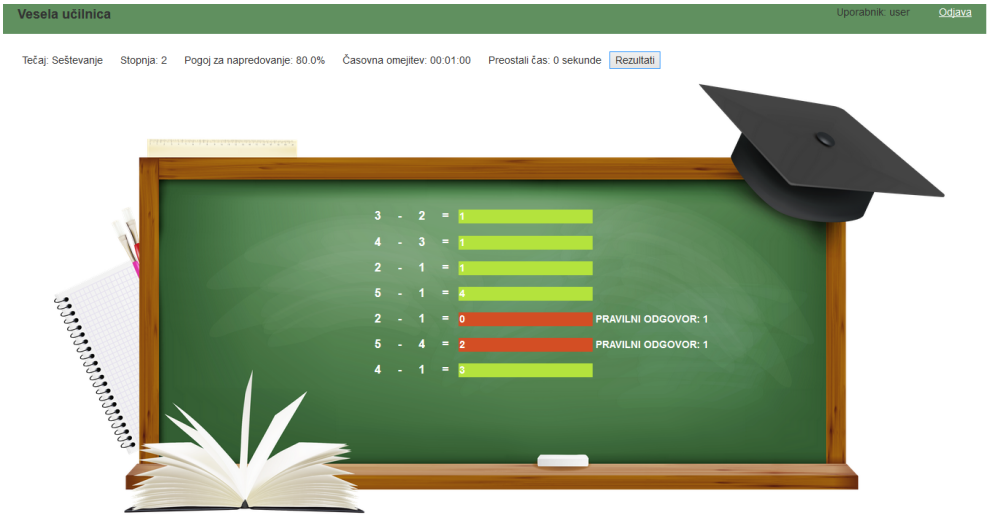
4.2.4 Prikaz rešitev

Po zaključku reševanja učenec dobi pregled rešenih nalog (slika 4.4). Reševanje je zaključeno ob izteku časa, ki je namenjeno za reševanje, učenec pa ima možnost predhodno zaključiti reševanje. Ob zaključku aplikacija za vsako nalogo preveri pravilen odgovor. Pravilni rezultat naloge se nahaja v objektu `Naloga`, ki smo ga shranili ob generiranju naloge. Pri napačnih odgovorih navedemo tudi pravilne rešitve.

4.2.5 Pregled lestvice

Učencu omogočimo pregled lestvice rezultatov za učno vsebino, za katero je tudi sam reševal preverjanje znanja.
Strežnik naredi poizvedbo o rezulatih na naslednji način:

```
1 public List<RezultatDTO> getRezultatiForLevel(Level level) {
2   TypedQuery<Rezultat> query = em.createQuery("Select r from
       Rezultat r where r.level=:lvl", Rezultat.class);
3   q.setParameter("lvl", level);
4   final List<Rezultat> resultList = q.getResultList();
```



Slika 4.4: Primer prikaza pravilnih in napačnih rešitev

```
5    List<RezultatDTO> rezultati = resultsToDto(resultList);
6    return rezultati;
7  }
```

Dobljene podatke prikažemo na odjemalcu. Za prikaz smo uporabili GWT razred `FlexTable`, ki omogoča dodajanje celic na zahtevo. Tako smo tabeli dodali vrstice šele, ko smo iz strežnika dobili odgovor, torej ob povratnem klicu.

4.3 Učitelj

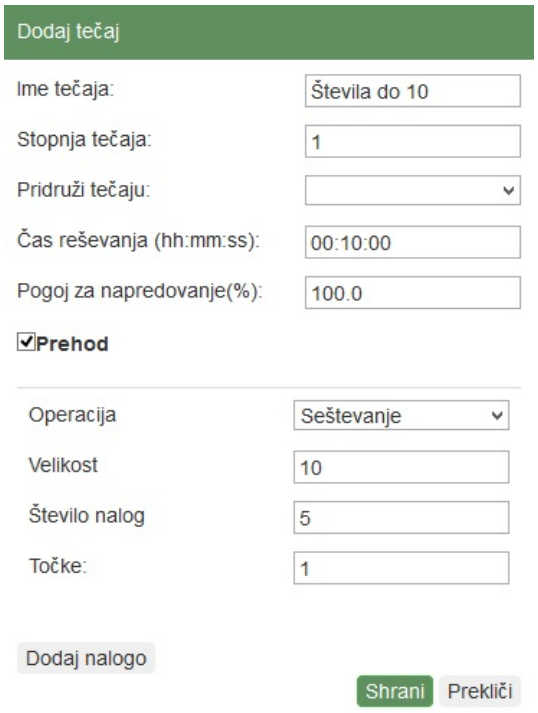
4.3.1 Dodajanje učne vsebine

Dodajanje učnih vsebin je pomembna funkcionalnost aplikacije, saj se na podlagi definicije učne vsebine kasneje generirajo naloge. Na meniju za dodajanje učitelj izpolni vnosna polja. Obvezna polja, ki jih je potrebno vnesti sta stopnja in pogoj za napredovanje. Če ti dve polji nista izpolnjeni, je uporabnik opozorjen. V primeru, da je polje s časovno omejitvijo prazno, sklepamo, da preverjanje znanja ne bo časovno omejeno. Prav tako učno vsebino, ki je ne pridružimo nobeni obstoječi, ustvarimo kot novo. Učna vsebina se po zaključku shrani v podatkovno

bazo in je na voljo za reševanje.

4.3.2 Urejanje učne vsebine

Urejanje učne vsebine izvedemo v istem pojavnem oknu, kot dodajanje (slika 4.5). Razlika je v tem, da za urejanje predhodno pridobimo vse informacije o izbrani učni vsebini in jih prikažemo v pojavnem oknu. Po končanem urejanju posodobimo podatke za to učno vsebino. Učne vsebine lahko tudi brišemo. Pred brisanjem se preveri ali ima izbrana učna vsebina stopnje. V tem primeru najprej izbrišemo vse stopnje in šele nato krovno učno vsebino. Preveri se tudi, ali za izbrano učno vsebino obstajajo rezultati. V tem primeru se uporabnika najprej obvesti in šele z njegovo potrditvijo se izbriše učna vsebina in njej pripadajoči rezultati.



Slika 4.5: Pojavno okno za dodajanje in urejanje učne vsebine

4.3.3 Pregled rezultatov

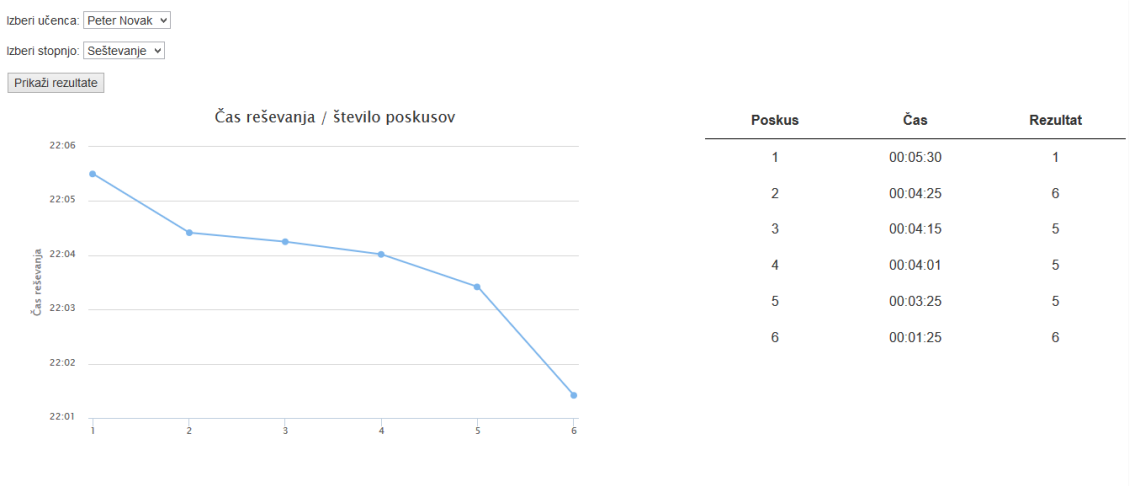
Učitelj ima na voljo različne statistike in pregled rezultatov. Pregled rezultatov za posameznega učenca omogoča pregled po učni vsebini, ki jih je reševal. Za grafični prikaz podatkov smo uporabili knjižnico GWT Highcharts [21]. Ta ponuja različne tipe grafov, ki so preprosti za uporabo.

Učitelj najprej izbere učenca, nato pa še učno vsebino, za katerega želi rezultate. Za izbrano kombinacijo aplikacija najprej pridobi podatke o rezultatih. Kot parametre podamo uporabnika in učno vsebino. Strežnik kot rezultat vrne seznam objektov tipa `RezultatDTO`.
Prvi graf predstavlja čas reševanja v odvisnosti od števila poskusov. Število poskusov nam predstavlja število objektov `RezultatDTO` v seznamu, ki so kronološko urejeni. Za vsak objekt se vzame čas reševanja. Par poskus-čas reševanja predstavlja točko na grafu.
Drugi graf predstavlja delež točnih odgovorov v odvisnosti od časa. Tudi v tem primeru za vsak objekt iz odgovora strežnika vzame rezultat in čas reševanja. Par čas-rezultat predstavlja točko na grafu. Primer prikaza rezultatov na grafu je prikazan na sliki 4.6.
Rezultate smo predstavili tudi v tabeli. Za prikaz v tabeli smo uporabili GWT razred `FlexTable`. Vsak zapis v tabeli predstavlja en poskus reševanja. Za vsak poskus so navedena še čas reševanja in število točnih odgovorov.

Učitelj ima možnost pregleda rezultatov za posamezno stopnjo brez določitve učenca. Tako dobi prikaz časa reševanja v odvisnosti od deleža pravilnih odgovorov vseh učencev za izbrano stopnjo. Ob postavitvi na točko na krivulji pa se pokažejo podatki učenca, kateremu pripada izbran rezultat. Pri poizvedbi rezultatov ne omejimo po uporabniku, ampak samo po učni vsebini.

4.3.4 Izvoz

Učitelj ima možnost izvoza generiranih nalog za izbrano učno vsebino, v formatu `pdf`. Generiranje `pdf` datoteke smo implementirali z uporabo knjižnice iText [22]. Za izbrano učno vsebino se na strani odjemalca zgenerirajo naloge (opredeljeno



Slika 4.6: Primer prikaza rezultatov na grafu in v tabeli

v 4.2.2) in se pošljejo na strežnik. Generiranje datoteke se izvede na strežniku.

Primer metode, ki zgenerira datoteko pdf:

```
1  public String createPdf() {
2     try {
3        String fileurl = "vzorec.pdf";
4        Document document = new Document();
5        PdfWriter.getInstance(document, new
             FileOutputStream(fileurl));
6        document.open();
7        Paragraph header = setHeader();
8        document.add(header);
9        setData(document);
10       document.add(content);
11       document.close();
12    } catch (Exception e) {
13       System.out.println("Err : " + e);
14    }
15       return "http://localhost:8080/UcniSistem/vzorec.pdf";
16    }
```

Primer metode, ki vsako zgnereirano nalogo pretvori v niz in jo doda na dokument:

```
1 public void setData(Document document) {
2   List<Naloga> naloge = getData();
3   for(Naloga naloga : naloge){
4     Paragraph paragraph = new Paragraph("n.getPrviOperator() +
          n.getOperacija() + n.getDrugiOperator() +
          Constants.Helper.ENAKO");
5     document.add(paragraph);
6   }
7 }
```

Ko je dokument pdf z nalogami ustvarjen, strežnik vrne odjemalcu pot za dostop do dokumenta. Na strani odjemalca nato prikažemo zgenerirano datoteko.

4.4 Skrbnik

4.4.1 Dodajanje novega uporabnika

Učitelj ali skrbnik lahko ustvarita novega uporabnika v sistemu. V uporabniški vmesnik vneseta ime in priimek uporabnika. Skrbnik pa mora še izbrati ali bo nov uporabnik imel vlogo učenca ali učitelja. Tako iz odjemalca na strežnik pošljemo ime, priimek in vlogo. Za uporabniško ime strežnik določi ime in prvo črko priimka. Trenutni datum in čas se pretvori v vrednost `long` oziroma milisekunde, vzame se zadnjih 5 znakov vrednosti in se jih doda uporabniškemu imenu.

Zgled generiranja uporabniškega imena:

```
Ime in priimek učenca:  Peter Novak
Trenutni datum in čas:  5.8.2016, 10:21:05
Pretvorjen časovni žig v long (ms):  1470385265000
Zgenerirano uporabniško ime:  petern65000
```

Strežnik nato najprej preveri ali uporabniško ime že obstaja. Če že obstaja ponovno zgenerira uporabniško ime. Če uporabniško ime še ne obstaja, zanj zgenerira

še geslo. Geslo je sestavljeno iz priimka in zadnjih 5 znakov iz `long` vrednosti trenutnega datuma in časa. Geslo se zakriptira z zgoščevalno funkcijo MD5 [23]. Podatki o uporabniku se nato shranijo v podatkovno bazo.

Poglavje 5

Analiza

Izdelana aplikacija predstavlja zanimivo in primerno okolje za uporabo v procesu izobraževanja. Izdelan učni sistem ne nadomešča učitelja, ampak ga lahko učitelj uporablja kot učni pripomoček. Ta pripomore k optimizaciji učenga procesa, saj učitelju ni potrebno sestavljati nalog in jih kasneje ocenjevati, učencem pa je učenje in reševanje nalog zabavnejše, saj aplikacija zaradi zanimivega uporabniškega vmesnika spominja na računalniško igro.

Za ovrednotenje razvitega učnega sistema smo izdelali analizo SWOT [24], kjer smo opredelili prednosti (angl. strenghts), slabosti (angl. weaknesses), priložnosti (angl. opportunities) in nevarnosti (angl. threats) aplikacije.

Prednosti

Prednost aplikacije so implementirane funkcionalnosti. Med njimi je zagotovo najpomembnejše avtomatsko generiranje nalog na podlagi prej definiranih učnih vsebin. Pomembna funkcionalnost so tudi rezultati preizkusov znanja, ki so tako zbrani na enem mestu. Prednost je tudi realizacija učnega sistema s spletnimi tehnologijami, ki omogočajo identifikacijo uporabnika in pa dostop do sistema iz različnih lokacij. Aplikacija predstavlja enostavno in hkrati učinkovito rešitev za sestavljanje nalog za utrjevanje ali preverjanje znanja ter avtomatsko preverjanje nalog po zaključku reševanja. Uporaba aplikacije je enostavna in primerna za uporabo v nižjih razredih osnovne šole, učiteljem pa predstavlja učinkovito orodje pri

izvajanju učnega procesa.

Slabosti

Slabost trenutno implementiranega učenga sistema je zagotovo omejen nabor funkcionalnosti, ki so na voljo uporabnikom. Slabosti so tudi zahteve za uporabo aplikacije. Uporabnikom mora biti omogočen dostop do uporabe računalnika, ta pa potrebuje še dostop do omrežja. Za uporabo aplikacije je potrebno še osnovno poznavanje uporabe računalnika, ki pa je lahko problem pri učencih v nižjih razredih osnovne šole.

Priložnosti

Implementiran učni sistem predstavlja osnovo za razvoj večjega učnega sistema, ki bi ponujal še druge funkcionalnosti in izboljšal proces izobraževanja. Aplikacijo lahko trenutno uporabljata učitelj in učenec. Učenec jo lahko uporablja neodvisno od tega, ali aplikacijo uporabljajo pri pouku, saj lahko na njej utrjuje znanje, npr. doma. Nabor vlog bi lahko razširili tako, da bi dodali še pomočnika učitelja ali asistenta, starše in druge osebe, ki so ponavadi vključeni v učni proces. Razširili bi lahko tudi nabor izobraževalnih vsebin. Trenutno je implementiran samo aritmetični modul, ki bi ga lahko nadomestimo s katerim drugim.

Nevarnosti

Nevarnost razvitega učnega sistema predstavlja predvsem oteženo zagotavljanje varnosit. Potrebno bi bilo izboljšati kriptiranje gesel, ki se sedaj kriptirajo samo z zgoščevalno funkcijo MD5.

Menim, da je aplikacija zanimiv učni pripomoček, ki lahko pripomore k izboljšanju izobraževanja. Zaradi zanimivega uporabniškega vmesnika, pripomore k večji motivaciji za utrjevanje znanja. Aplikacija je primerna za uporabo, saj pospeši sam učni proces. Z uporabo aplikacije učitelj za ustvarjanje učnih vsebin le določi parametre, kar navadno traja nekaj minut, aplikacija pa potem na podlagi teh parametrov zgenerira naključne naloge. S pričetkom novega šolskega leta nameravamo tudi izvesti testiranje aplikacije na eni izmed osnovnih šol.

Poglavje 6

Sklepne ugotovitve

V diplomskem delu smo se spoznali z delovanjem in principi učnih sitemov. Pregledali in analizirali smo obstoječe sisteme ter jih med seboj primerjali. Ugotovili smo, da se tovrstni sistemi pogosto uporabljajo, saj omogočajo učenje na daljavo. Ponujajo se številne funkcionalnosti, ki so izboljšale tradicionalni način poučevanja v učilnicah. Najpomembnejše izmed njih so zagotovo:

- avtomatizirano preverjanje znanja in ocenjevanje učenca,
- dostopnost gradiv, ki so zbrana na enem mestu,
- pregled aktivnosti učenca.

Ideja diplomske naloge je bila izdelati okolje, ki bo zanimivo in predvsem uporabno. Odločili smo se za razvoj prototipa preprostega učnega sistema, ki implementira naslednje principe:

- upravljanje učnih vsebin,
- posredovanje učnih vsebin,
- preverjanje znanja,
- generiranje nalog.

Implementiran prototip učnega sistema predstavlja učinkovit pripomoček pri izvajanju izobraževalnega procesa, saj učencem povečuje motivacijo za učenje, učiteljem pa olajša in pospeši izvajanje učnega procesa. Ogrodje aplikacije je zasnovano na način, da je mogoče izvajati učne procese z različnimi vsebinami. Sami smo se osredotočili na poučevanje aritmetike v nižjih razredih osnovne šole. Aritmetični modul bi lahko zamenjali s katerim drugim, saj mehanizmi za prikaz učnih vsebin, ocenjevanje in prikaz rezultatov niso vezani na vsebino. Parametrizirano generiranje kvizov omogoča, da učitelj prilagaja učni proces učencem, njihovemu znanju in zanimanju.

Učni sistem dopušča še druge možnosti za nadgradnjo. Sistem bi vsekakor lahko izboljšali z razširitvijo vlog in z dodatnimi funkcionalnostmi, kot je naprimer dinamično odzivanje sistema na aktivnosti učenca. Sistem bi tako usmerjal potek izobraževanja na način, da bi ugotovil, na katerih področjih je znanje učenca šibkejše in mu ponujal več učnih vsebin iz tega področja. Dodatna funkcionalnost bi bila tudi pomoč učencu. Realizirana bi bila v obliki žetonov, ki jih učenec pridobi s pravilnimi odgovori. Zaradi razširjenosti mobilnih naprav bi lahko implementirali mobilno aplikacijo, ki bi vsebovala funkcionalnosti sistema in omogočili, da je dostop neodvisen od znanja uporabe računalnika, hkrati pa bi postavili novo zahtevo, kot je uporaba in dostop do mobilne naprave.

Literatura

[1] B. Holmes, J. Gardner, "E-learning: Concepts and practice", Sage, 2006.

[2] Epignosis. "E-LEARNING. CONCEPTS, TRENDS, APPLICATIONS", 2014.

[3] W. R. Watson, S. L. Watson, "What are learning management systems, what are they not, and what should they become?", *TechTrends*, št. 51, zv. 2, str. 28, 2007.

[4] About Moodle. [Online]. Dosegljivo: https://docs.moodle.org/31/en/About_Moodle. [Dostopano 30. 7. 2016].

[5] Moodle Features. [Online]. Dosegljivo: https://docs.moodle.org/31/en/Features. [Dostopano 30. 7. 2016].

[6] Edmodo Platform Overview. [Online]. Dosegljivo: https://developers.edmodo.com/resources/overview/#api-features. [Dostopano 30. 7. 2016].

[7] A. McAuley, et al. "The MOOC model for digital practice.", 2010.

[8] Welcome to coursera. [Online]. Dosegljivo: http://academicinnovation.wvu.edu/MOOC/PDF/Coursera.pdf [Dostopano 1. 8. 2016].

[9] L. Breslow, et al., "Studying learning in the worldwide classroom: Research into edX's first MOOC.", *Research & Practice in Assessment*, št. 8, 2013.

[10] I. Padayachee, "Intelligent tutoring systems: Architecture and characteristics", *Proceedings of the 32nd Annual SACLA Conference*, 2002.

[11] Introduction to Java Platform, Enterprise Edition 7. [Online]. Dosegljivo: http://www.oracle.com/technetwork/java/javaee/javaee7-whitepaper-1956203.pdf [Dostopano 1. 8. 2016].

[12] Developing Enterprise JavaBeans. [Online]. Dosegljivo: https://docs.oracle.com/cd/E13226_01/workshop/docs92/pages/pdf/ejb.pdf [Dostopano 1. 8. 2016].

[13] About wildfly. [Online]. Dosegljivo: http://wildfly.org/about/ [Dostopano 1. 8. 2016].

[14] About postgreSQL. [Online]. Dosegljivo: https://www.postgresql.org/about/ [Dostopano 1. 8. 2016].

[15] GWT overview. [Online]. Dosegljivo: http://www.gwtproject.org/overview.html [Dostopano 1. 8. 2016].

[16] CSS. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Cascading_Style_Sheets [Dostopano 1. 8. 2016].

[17] H. S. Oluwatosin, "Client-server model", *IOSR Journals (IOSR Journal of Computer Engineering)*, št. 16, str. 57-71, 2014.

[18] Ajax Communication: Introduction. [Online]. Dosegljivo: http://www.gwtproject.org/doc/latest/tutorial/clientserver.html [Dostopano 1. 8. 2016].

[19] Using GWT RPC. [Online]. Dosegljivo: http://www.gwtproject.org/doc/latest/tutorial/RPC.html [Dostopano 1. 8. 2016].

[20] Using JBoss Login Modules. [Online]. Dosegljivo: https://docs.jboss.org/jbosssecurity/docs/6.0/security_guide/html/Login_Modules.-html [Dostopano 1. 8. 2016].

[21] GWT Highcharts. [Online]. Dosegljivo: http://www.moxiegroup.com/moxieapps/gwt-highcharts/ [Dostopano 1. 8. 2016].

[22] IText. [Online]. Dosegljivo: http://itextpdf.com/ [Dostopano 1. 8. 2016].

[23] The MD5 Message-Digest Algorithm. [Online]. Dosegljivo: https://www.ietf.org/rfc/rfc1321.txt [Dostopano 1. 8. 2016].

[24] W. Gretzky. "Strategic Planning And SWOT Analysis.", 2010.